Week 3

Q1.

	Mean execution time	Standard deviation
Base	2.33 ms	56 µs
Iterrows Haversine	1.76 ms	248 µs
Apply	7.01 ms	2.63 ms
Vectorized (Pandas series)	6.1 ms	123 µs
Vectorized (Numpy arrays)	3.09 ms	450 µs
Cythonized loop (unaltered)	1.03 ms	33.7 µs
Redefined with C	656 µs	23.1 µs

We can observe that the base function with just looping is more effective than some other methods because of the small dataset. Using the Apply method is slowest because Pandas .apply() is not optimized for large numerical computations. Vectorized (Pandas series) is faster than .apply() method but Pandas is still slower than NumPy for math-heavy operations. Cythonized loop is Much faster due to Cython compiling Python loops into C for efficiency. Finally for more efficiency we used C libraries which gave the lowest mean execution time.

Q2.

	Minimum execution time	Mean execution time	Maximum execution time
Base	3.09 s	3.63 s	3.97 s
Vectorized using matrix operations	24.4 µs	41.85 µs	449.5 µs
Vectorized using Apply	335.9 µs	552.9 µs	1293.6 µs

The brute-force methods functioned well in Python until I implemented vectorized approaches. Execution speeds showed that R processed vectorized commands faster than any other R operation due to its optimized structure for big data analysis. Matrix operations within R outperformed even optimized Python code written in Cythonic for speed execution. R demonstrates an inherent capability to perform high-performance computations on large-scale data especially during statistical and mathematical operations.

Q3.

Based on my experience with R, faster execution times are achievable when using well-optimized mathematical functions. However, this may not always be true in Python due to differences in how the two languages handle computations internally. The main challenge I faced with R was that it took significantly longer to write and debug the code, as I encountered numerous errors during execution. In contrast, Python made coding much easier, as many small details are handled by built-in functions, allowing for faster development. In my opinion, for professional projects requiring high-performance computations, I would choose R, whereas for personal research or projects, I would prefer Python due to its ease of use and flexibility.

Q4.

The selection between Python and R as programming languages depends on multiple factors including user environments in addition to coding simplicity and run-time performance. Machine learning requires a vast number of libraries which Python provides along with general-purpose programming tools that make this programming language a leading tech industry standard. R proves most suitable for research environments when statistical analysis and visualization tasks must be performed because of its strong capabilities in this area. The industrial sector heavily relies on Python for its applications yet R continues to dominate statistical data analysis particularly in medical research. My choice between Python and R depends on my targeted line of work and the requirements of my data analysis tasks. While I focus on enhancing my skills in both languages I prefer Python due to its user-friendly nature.